Integrazione di Sistemi Embedded
Laboratorio 04

Matteo Perotti 251453
Giuseppe Puletto
Luca Romani 255244
Giuseppe Sarda 255648

November 23, 2018

1 Introduzione

Il quarto laboratorio ha come obiettivo l'esplorazione delle possibilità che il comando **make** può fornire al programmatore durante la fase di test di un determinato progetto. Strumento fondamentale per questa fase è il file **Makefile**.

2 Approccio alla scrittura del Makefile

Per la scrittura del Makefile, il quale contiene tutte le "ricette" necessarie per la completa compilazione di un progetto, si è fatto fede a quanto spiegato dal professore durante la lezione e si è consultato il manuale GNU relativo al comando *make* reperibile alla pagina web GNU make.
Di seguito sono riportate le informazioni più rilevanti ricavate dal manuale, alcune delle quali sono state usate all'interno dei Makefile prodotti.

Phony targets

Usando il "token" **.PHONY** si riesce ad indicare un target che non corrisponde al nome di un vero e proprio file, ma ad una ricetta eseguita soltanto se espressamente richiesto. Questa direttiva è utile, inoltre, in caso di omonimia tra un taget e un file, i quali però non sono logicamente collegati, e che quindi, senza un'indicazione specifica, sarebbero erroneamente vincolati. Dichiarando dunque:

```
.PHONY: clean
clean:
        #do something
```

il target *clean* viene eseguito esclusivamente quando richiesto all'esecuzione del comando make evitando eventuali malfunzionamenti nel caso esita un file nominato "clean" nella directory del progetto.

3 Impementazione del test

Per la realizzazione del target test si deciso di dividere la verifica del codice scritto per il precedente laboratorio in 4 parti:

- Creazione di uno script Python in grado di generare comandi di disegno casuali sotto specifiche
- Scrittura di un codice Python, basato sugli identici algoritmi usati durante il precedente lab, che si comportasse secondo specifiche relativamente alla parte funzionale ma che lasciasse totale libertà implementativa
- Esecuzione dei comandi generati al punto uno da parte del codice C e dello script Python
- Confronto dell'output finale delle due esecuzioni

Si noti che i limiti di questo test stanno nel fatto che, nel caso in cui gli algoritmi di disegno fossero sbagliati in partenza, non c'è modo di accorgersene. I casi limite inoltre non vengono così testati perché i comandi al punto 1 vengono per definizione prodotti al fine di rispettare le specifiche.

Generazione dei comandi

Lo script per la generazione dei comandi accetta dall'utente un numero di comandi totali che crea usando la funzione *randint* contenuta nel modulo *random*. Infine i comandi generati vengono scritti all'interno di un file che verrà poi passato ai file eseguibile. L'esecuzione finisce con la scrittura di un **token** che segnala la fine dello strem di comandi.

Script di test

Lo script di test che produce il risultato esatto dall'esecuzione dei comandi, come già specificato in precedenza, si basa sugli stessi algoritmi di riferimento del codice C. Genera e modifica una matrice di interi più facilmente accedibile e modificabile che della struttura usata nel codice originale. Ulteriori vantaggi dell'utilizzo del python sono dovuti alla completa assenza di interfacce e libertà di gestione della realizzazione del codice.